METHOD TO WRITE IN A NON VOLATILE MEMORY AND SYSTEM TO IMPLEMENT SUCH METHOD

This invention concerns a method to write in a non volatile memory of an electronic assembly such as for example an onboard system. More precisely, the objective of this invention is to propose a method to optimise the time to write in this type of memory.

The invention also concerns an onboard system for the implementation of such a method.

The invention applies more especially to a smart card.

In the context of the invention, the term "onboard system" must be taken in its broadest sense. It concerns in particular all types of light terminals equipped with an electronic chip and more especially the smart cards as such. The electronic chip is itself equipped with information processing means (for example a microprocessor) and information storage means.

TECHNICAL FIELD

Writing permanent data in a non volatile memory of an onboard system generally consists in a succession of erase/programming steps of said memory. Erasure consists in switching to "low" state (referred to later as '0') all memory cells of a specific region (called "block" or "page"). Programming consists in switching to "high" state (written '1') only part of said specific region. Writing consists in erasure of a region and programming suitable bits in said region.

On the present cards, the non volatile memory uses EEPROM technology. Write operations in EEPROM memory are very slow, about 4 ms. The erasure and programming times are similar, about half the write time i.e. approximately 2 ms. Consequently, the latencies induced by the writes in memory hide the true performance of the processor.

Currently, a new non volatile memory technology is emerging in smart cards: Flash technology. Flash technology differs from EEPROM technology especially as regards the significantly different characteristics of programming and erasure. In a Flash memory therefore, there is a large dissymmetry between the time required for programming, which is quite fast, and the time required to erase a previously programmed cell, identical to the time required for erasure in EEPROM memory. For example, the time required for programming may reach 10 μs (for a small amount of memory).

One objective of the invention is to optimise the write times in a non volatile memory of an electronic assembly equipped with Flash type memory.

Another objective of this invention is to propose a solution which could be implemented in an onboard system.

SUMMARY OF THE INVENTION

This invention concerns a method to write in a Flash type memory of an electronic module characterised in that it consists in associating at least two physical areas of said memory, called mirror areas, with the same logical area and during a write in said logical area, in programming the content of said logical area in one of said blank mirror areas.

This invention also concerns the electronic module comprising information processing means, a FLASH type non volatile memory characterised in that it comprises a mirror memory formed from at least two physical areas and associated with the same logical area, each new programming operation in said logical area taking place in an area of the blank mirror memory as well as the card in which said module is integrated.

BRIEF DESCRIPTION OF THE DRAWINGS

Other purposes, features and advantages of the invention will appear

on reading the description which follows of the implementation of the method according to the invention and of a mode of realisation of an electronic system designed for this implementation, given as a non-limiting example, and referring to the attached drawings in which:

- figure 1 is a schematic view of an example of realisation of an electronic module integrated in a portable object such as a smart card;
- figure 2 is a schematic view of the steps of the method according to this invention;
- figure 3 is a schematic view of a first mode of realisation of association between logical and physical areas in the method according to this invention;
- figure 4 is a schematic view of a second mode of realisation of association between logical and physical areas in the method according to this invention;
- figures 5a to 5c are schematic views of the various types of write in the Flash type memory;
- figure 6 is a schematic view of a first mode of realisation of erasure and regeneration of physical areas in the method according to this invention;
- figures 7a to 7c are schematic views of a second mode of realisation of erasure and regeneration of physical areas in the method according to this invention.

WAY OF REALISING THE INVENTION

The method according to the invention aims to optimise the write time in a memory of an electronic assembly, and for example an onboard system such as a smart card, any portable object equipped with an electronic module. The electronic assembly includes at least a processor and a Flash type non volatile memory. In the following description, FLASH type memory means any non volatile memory displaying dissymmetry between the time

required for programming and erasure.

Without this limiting its scope in any way whatsoever, the preferred application of the invention will be discussed below, i.e. applications based on integrated circuit cards.

Cards with integrated circuit also called smart cards are small plastic devices, which contain one or more embedded integrated circuits. A card with integrated circuit can be for example a memory card or a microprocessor card called also microprocessor chip card.

In a particular embodiment of the present invention shown in figure 1, the smart card 1 contains an integrated electronic unit 2: the electronic unit 2 comprises at least a microprocessor CPU 3 with two-way connection via an internal bus 5 to a non volatile memory 7 of type Flash storing at least a program to be executed, a volatile memory 11 of type RAM and input/output means 13 to communicate with the exterior. The unit 2 may comprise additional components not shown, connected to the internal bus. This type of unit is generally manufactured as a monolithic integrated electronic circuit, or chip, which once physically protected by any known means can be assembled on the integrated circuit card or similar for use in various fields, such as the bank and/or electronic payment cards, mobile radio telephony, pay television, health and transport.

This invention consists in a software method in order to benefit from the dissymmetry of the programming/erasure times of a non volatile memory, especially FLASH, to optimise the write times in non volatile memory of a smart card. A "mirror" memory is therefore defined and divided into n physical areas designed to contain the same logical area of the program.

Figure 1 shows an example of mirror memory mechanism.

With the system in its initial state, all the mirror memory areas are blank, i.e. empty, ready to receive and store data. When the program wants to make a write E1 in the logical area ZL, it does so by programming (fast) the first physical area ZP1. ZP1 is the so-called active or current physical area in which the content of the logical area must be read. During the next write E2 on this logical area ZL, we avoid erasing (slow) the first

physical area ZP1 by programming in the second physical area ZP2 (still blank). Area ZP2 becomes the active area. This method can be repeated until the mirror memory is saturated (or until the system finds a convenient time to erase the physical areas used, as will be seen below).

In order to re-use all physical areas, the mirror memory must be periodically erased. Erasure can be carried out at any time convenient for the system, and this erasure can benefit from the "block mode" of FLASH memories. The erasure of these physical areas is in fact optimised firstly by erasing all areas in a single operation and secondly by carrying out the erasure in a way that does not block the system.

A first method, known as "time multiplexing", is a purely software realisation. Erasure is carried out by the card system when the system is waiting, especially for an external event such as a command from the terminal. A second method, known as "space multiplexing" requires a hardware support to execute concurrent tasks. The erasure task is in fact launched by the card system and executed in parallel with normal program execution. This second implementation will preferably be carried out either using a bi-port FLASH memory or using a bi-bank FLASH memory.

In short, separating the programming/erasure cycle described by the invention offers the advantage of fast programming in FLASH memory and optimises memory erasure operations. The invention therefore reaches a compromise between memory use and performance.

Several modes of realisation of the invention are described below, in three sections:

Section 1: Realisation of association between logical area/physical areas.

Section 2: Area write algorithm.

Section 3: Erasure and regeneration of physical areas.

Modes of realisation of the association between logical area/physical areas (Section 1) are described below. In order to make the association between logical area/physical areas, we need to know the active

physical area (the current "mirror" area in which the content of the logical area must be read). It must be possible to quickly modify this data when changing physical area, to avoid penalising the programming operations. The data must therefore be stored in RAM or in a previously erased FLASH area.

A first realisation consists in a simple RAM counter, associated with the logical area, containing the number of the active area. The area is changed by incrementing the counter. When the card is initialised or in case of tearing, the physical areas are scanned to determine the number of areas "used" Zpu, i.e. the areas in which the content of the associated logical area at a given time has been programmed and not yet erased. The counter is initialised with this value.

Figure 2 illustrates a write operation requiring a change of active physical area for this first realisation.

A second realisation consists in a bit field in FLASH, associated with the logical area. Each bit represents the use state of a physical area ('1' → used; '0' → blank). The change of physical area is carried out by programming the bit corresponding to the newly active blank area. The complete bit field is erased when all physical areas are regenerated. For example, the active area may be determined as being the least significant area used in the bit field.

Figure 3 illustrates a write operation requiring a change of active physical area for this second realisation.

Modes of realisation of the write algorithm for an area (section 2) are described below, referring to figures 4a to 4c which illustrate the various comparison operations. The active physical area on the left contains in bold the bits to be modified. The new active physical area (the same as the old area in figure 4a) and the bits actually programmed, in bold, are shown on the right.

In the simplest approach, writing the entire logical area involves using a new physical area, whereas a partial write of the logical area involves reading in the current physical area, replacing the appropriate portion then

rewriting in a new physical area. This operation can be optimised by determining whether the current physical area can be re-used.

The method consists in first reading the current area and comparing it with the portion to be written.

- If the two contents are identical, nothing is written and the active physical area remains the same (figure 4a).

- If only bit programming operations are required (i.e. switch from '0' to '1'), the active area is not changed and the corresponding bits are programmed in the current area (figure 4b).

- Otherwise, the current area is read and masked by the portion to be written, then everything is programmed in a new active area (figure 4c).

Note that reading the current area beforehand does not have a significant impact on the performance of the method, since reads in non volatile memory are fast, just a few processor cycles. In addition, the content of the current area can be stored temporarily in RAM (which then acts as cache memory).

In a variant of the method (figure 4c) described above, the area is not entirely programmed, but just the portion which is actually different (greyed in the figure). Although this method involves more complex management, it may be better either because there is a significant gain when programming the non volatile memory or because the bit programming time is very high.

Modes of realisation of the regeneration of the physical areas (section 3) referring to figure 5 are described below.

"Time multiplexing" consists in separating the programming/erasure operations in time. In normal operation, the system only carries out programming. When it becomes inactive (or when all the areas are full), it erases them and is blocked during this period. For example, reception of a command on the I/O line of a smart card can be long (several hundred ms), the system takes advantage of this time to trigger an erase.

A purely software mechanism to erase the areas (figure 5) consists in copying the active physical area (the "mirror") in a buffer area, then in erasing all mirror physical areas and lastly in copying the buffer into the first physical area available. This mechanism is illustrated in the following diagram.

"Space multiplexing" consists in carrying out in parallel the erase operation and the programming/read operations on the logical area. Bi-bank FLASH is used to carry out this multiplexing. The read/programming/erase operations are generally exclusive on a FLASH, in particular it is impossible to erase one memory area while programming or reading another. The bi-bank FLASH has two banks on which operations can be carried out in parallel (even though each bank has the same constraints as the traditional FLASH).

The realisation on this memory assumes that the logical area has at least one "mirror" area in each bank. The bank containing the active area is used for the programming and the read, whilst the mirror areas in the other bank are completely erased (if possible) at the same time. The system changes active bank when all the mirror areas of the bank have been used. Figures 6a to 6c illustrate this realisation.

On figure 6a, the programming/read operations are carried out on bank A whilst bank B is erased.

On figure 6b, B is erased, the system continues to work on A until the physical areas are saturated.

On figure 6c, when A reaches saturation, B becomes the active bank and the system erases A in parallel.